CI/CD 持续集成．持续交付项目

[1. Gitlab服务器的搭建 1](#_Toc1253244555)

[1.1. Gitlab服务器的准备环境: 1](#_Toc355872260)

[1.2. 使用docker容器服务安装gitlab服务． 2](#_Toc1430339793)

[1.2.1. 准备gitlab的中文包 2](#_Toc1926665030)

[1.2.2. 挂载docker软件的镜像,并修改git服务器的yum厂库． 2](#_Toc1054999999)

[1.2.3. 安装docker服务,并启动docker服务． 2](#_Toc409622907)

[1.2.4. 导入中文版gitlab镜像 3](#_Toc1767399766)

[1.2.5. 将物理主机ssh端口改为2022后,起动容器 3](#_Toc1012022147)

[1.2.6. 浏览器登陆并设置新的密码，并设置项目，用户．组 3](#_Toc527678212)

[1.2.7. 主程序员推送代码测试 5](#_Toc1110440222)

[2. 搭建jenkins服务 7](#_Toc316480093)

[2.1. 安装jenkins的依赖包java和jenkins 7](#_Toc1243671007)

[2.2. 启动服务，并查看 7](#_Toc1120549807)

[2.3. 初始化Jenkins 7](#_Toc1126841208)

[2.3.1. 初始化 7](#_Toc1138964363)

[2.3.2. 选择插件来安装 8](#_Toc7879821)

[2.3.3. 选择无来安装 8](#_Toc1871312929)

[2.3.4. 选择用admin用户 9](#_Toc968892122)

[2.3.5. 登陆并修改admin密码1234.com 9](#_Toc338388917)

[2.3.6. 安装chinese中文插件，Git Parameter 10](#_Toc209348270)

[2.3.7. 在jenkins创建一个项目 10](#_Toc1011086881)

[2.3.8. 检查jenkins与gitlab服务器是否关联成功． 11](#_Toc2053109803)

[2.4. 构建分发服务器管理 12](#_Toc1896672759)

[2.4.1. Jenkins服务器上安装git和web网站 12](#_Toc1141930574)

[2.4.2. 修改工程构建工程(续1) 13](#_Toc311885892)

[2.4.3. 构建测试并查看jenkins服务上 15](#_Toc200912367)

[2.4.4. 创建版本文件(版本文件的属组是jenkins) 15](#_Toc57371191)

[2.4.5. 服务器测试． 16](#_Toc1326939191)

[2.4.6. 网站一键部署代码编写． 17](#_Toc630967977)

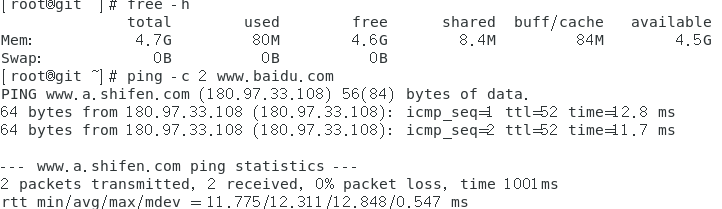
[2.4.7. 网站网页代码更新使用方案 19](#_Toc165741806)

[2.4.8. 网站网页代码应急方案 20](#_Toc1295871135)

# Gitlab服务器的搭建

## Gitlab服务器的准备环境:

准备换机一台内存大于4Ｇ的服务器，并系统在redhat6.5以上.安装docker服务，并要求该服务器的80/443/22端口不能被占用，所以需要修改ssh的端口．



## 使用docker容器服务安装gitlab服务．

### 准备gitlab的中文包

### 挂载docker软件的镜像,并修改git服务器的yum厂库．

（/linux-soft/04/openstack/RHEL7-extras.iso)

[root@room9pc01]#mount RHEL7-extras.iso /var/ftp/extras/

[root@git ~]#cat /etc/yum.repos.d/local.repo

[local\_repo]

name=CentOS-$releasever - Base

baseurl="ftp://192.168.1.254/centos-1804"

enabled=1

gpgcheck=0

[extras]

name =docker

baseurl =ftp://192.168.1.254/extras

enabled=1

gpgcheck=0

### 安装docker服务,并启动docker服务．

[root@git ~]#yum -y install docker

[root@git ~]#systemctl start docker

### 导入中文版gitlab镜像

[root@git ~]# docker load < gitlab\_zh.tar

### 将物理主机ssh端口改为2022后,起动容器

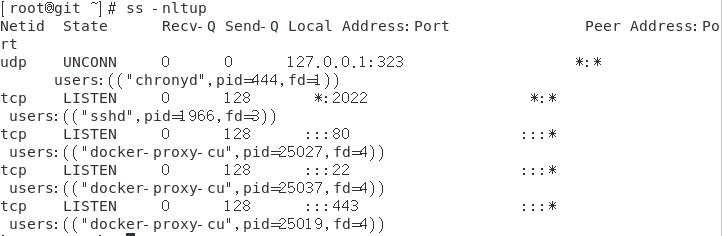
[root@git ~]# docker run -d -h gitlab --name gitlab -p443:443 -p 80:80 -p22:22 --restart always -v /srv/gitlab/config:/etc/gitlab -v /srv/gitlab/logs:/var/log/gitlab -v /srv/gitlab/data:/var/opt/gitlab gitlab\_zh:latest

1. **查看容器的启动状态．及服务端口**

docker ps

# 当状态中，显示healthy时，容器才能正常使用

[root@git ~]# ss -nltup



### 浏览器登陆并设置新的密码，并设置项目，用户．组

管理员用户名:root 密码:Admin@123

项目名称：myweb

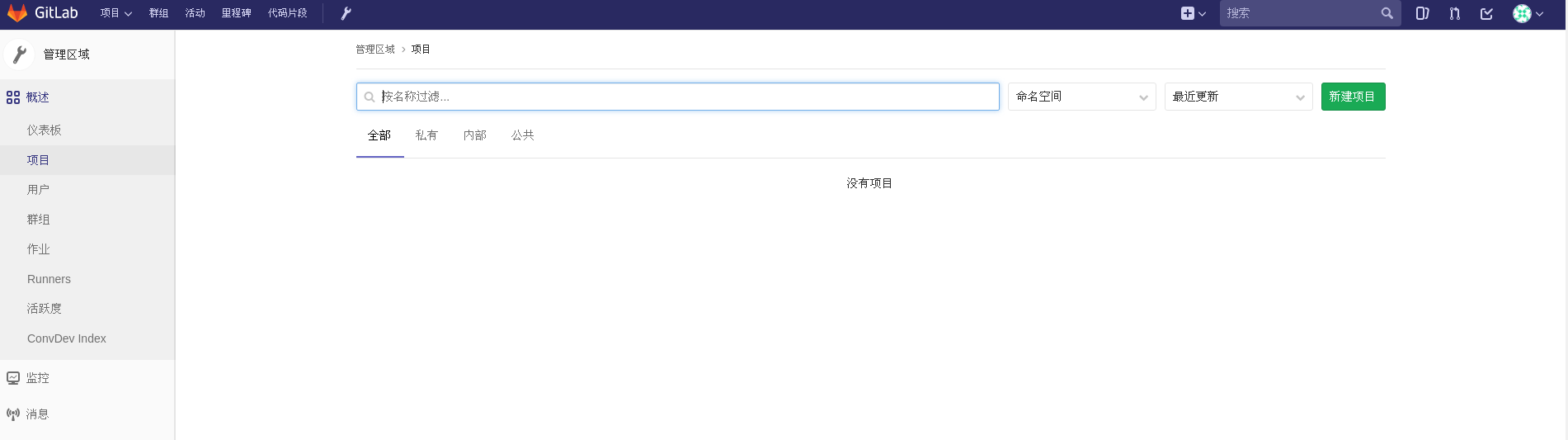
组名称：devops

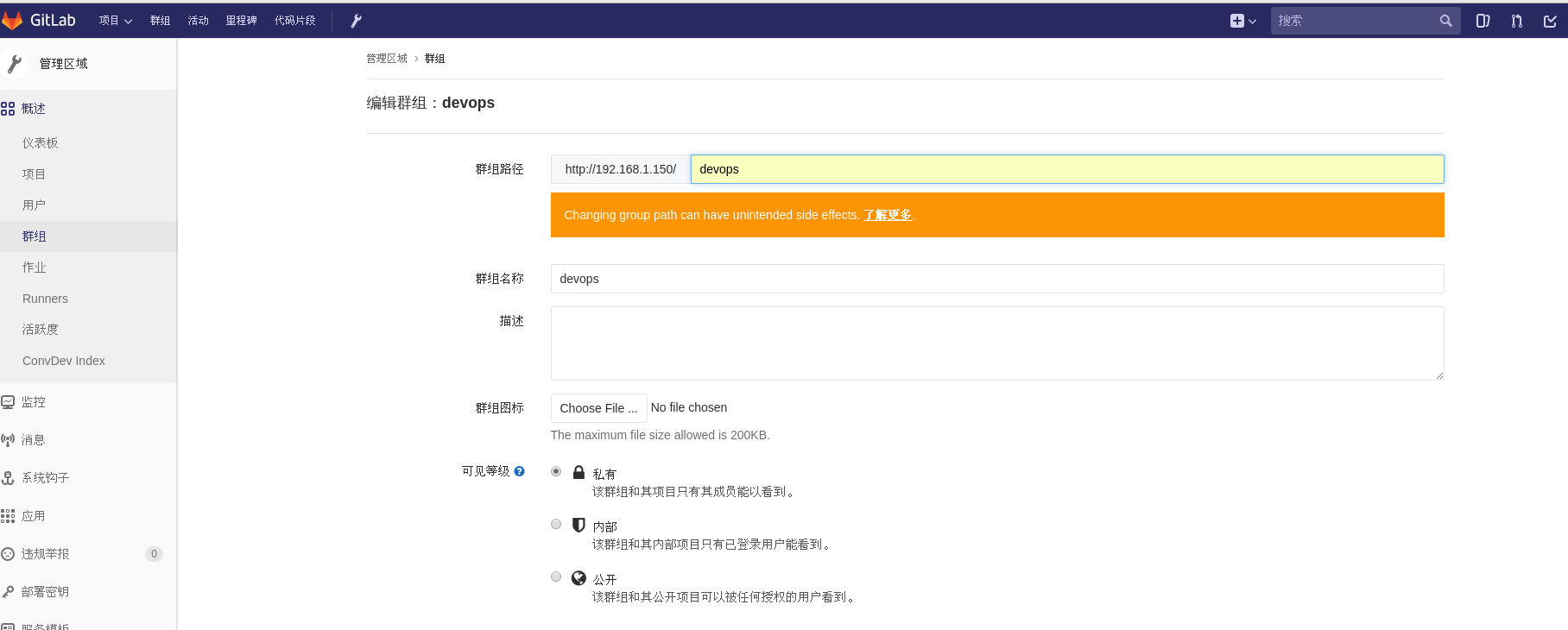
用户名称：zh

1. **新建普通用户： 用户名为：zh 密码:1234.coM**

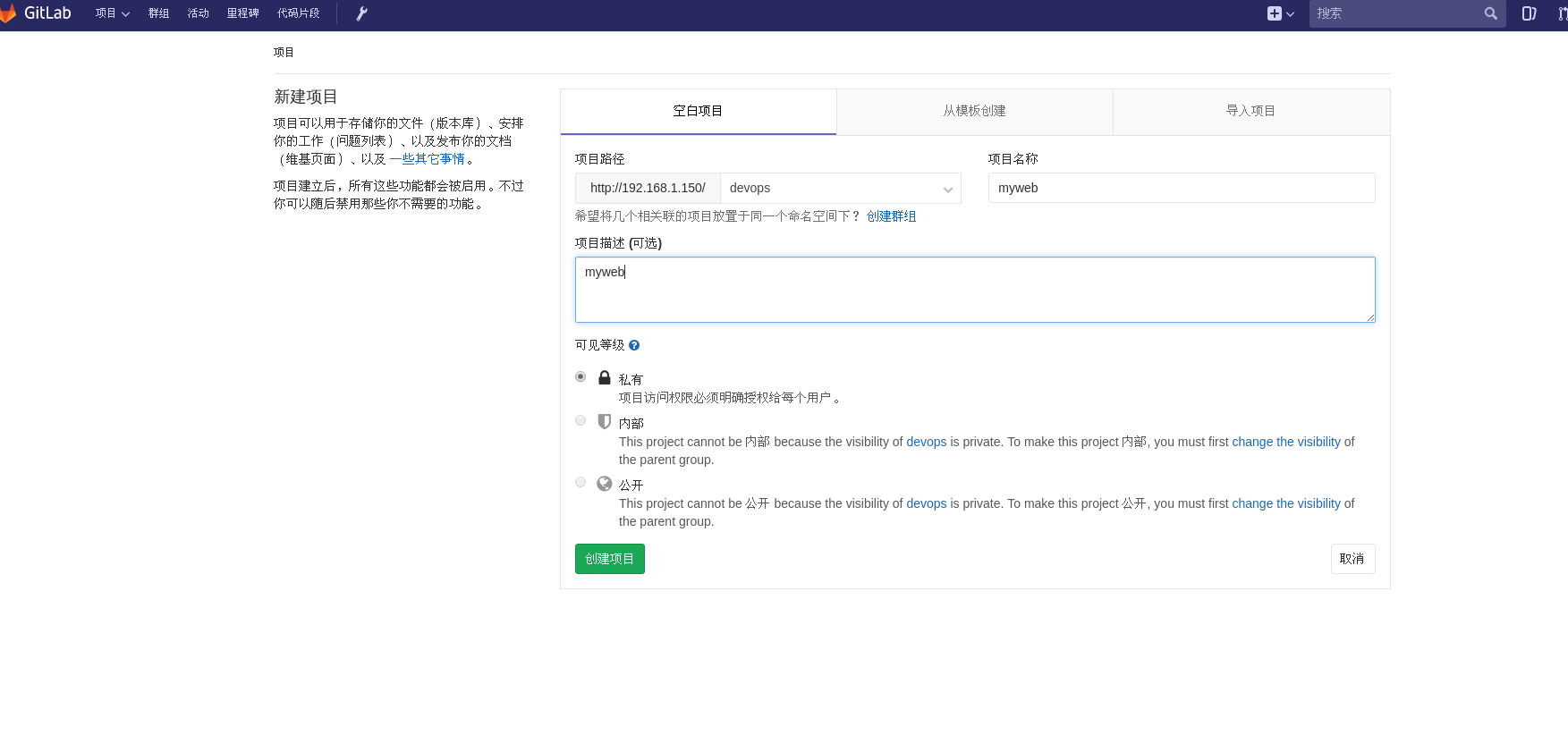


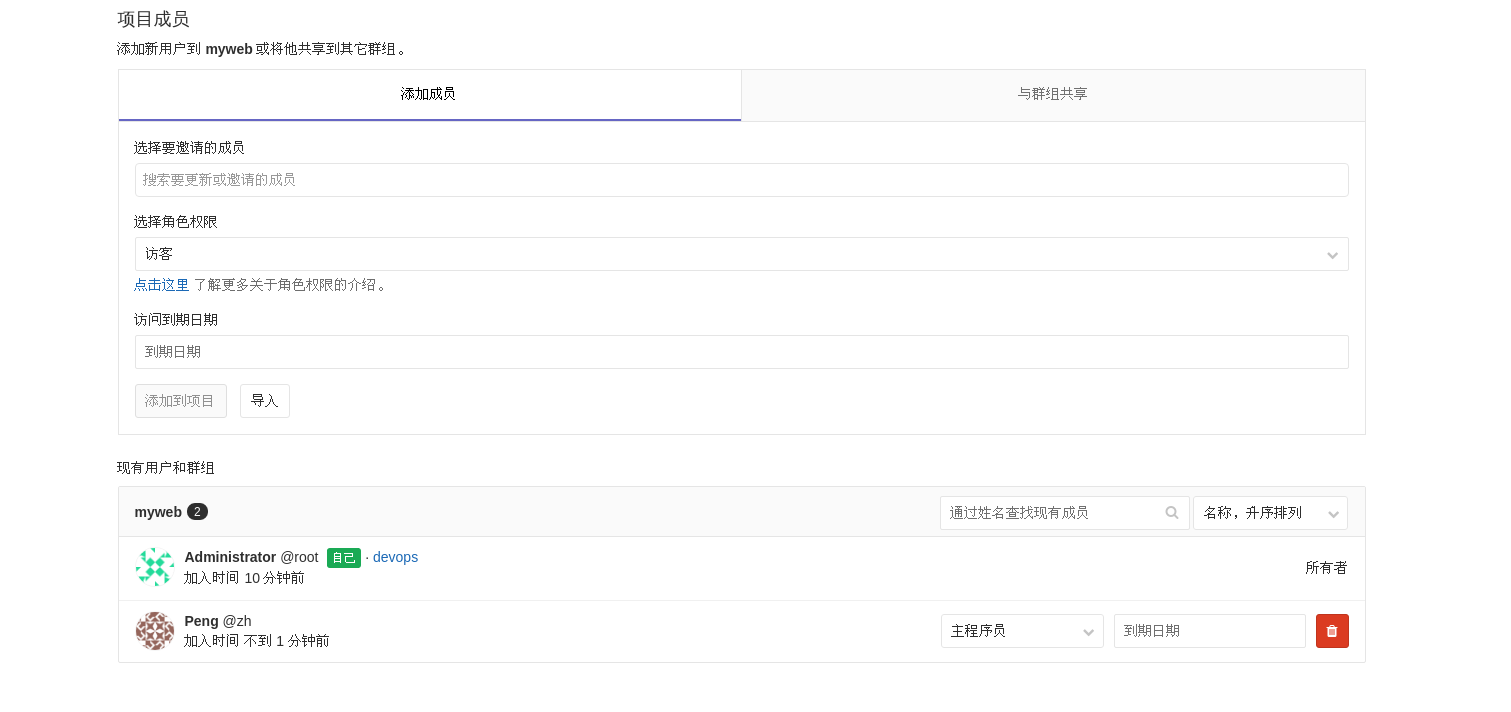
1. **新建公共组**





1. **新建公共项目，并添加主程序员用户**





### 主程序员推送代码测试

1. **安装git**

[root@gcs-office ~]# yum -y install git

1. **Git 全局设置**

[root@gcs-office myweb]# git config --global user.name "Peng"

[root@gcs-office myweb]# git config --global user.email ["942803514@qq.com"](mailto:\"942803514@qq.com\")

[root@gcs-office myweb]# git config --global core.editor vim

[root@gcs-office myweb]# git config --list

user.name=Peng

user.email=942803514@qq.com

core.editor=vim

core.repositoryformatversion=0

core.filemode=true

core.bare=false

core.logallrefupdates=true

[root@gcs-office myweb]# cat ~/.gitconfig

[user]

name = Peng

email = 942803514@qq.com

[core]

editor = vim

1. **创建新版本库**

[root@gcs-office ~]# mkdir myweb

[root@gcs-office ~]# cd myweb/

[root@gcs-office ~]# git init .

[root@gcs-office myweb]# ls -a

. .. .git index.html

1. **上传代码**

[root@gcs-office myweb]# git add .

[root@gcs-office myweb]# git commit -m "add web file 1.0"

[root@gcs-office myweb]# git tag 1.0　(打tag是为了后面jenkins中使用)

[root@gcs-office myweb]# git remote rename origin old-origin

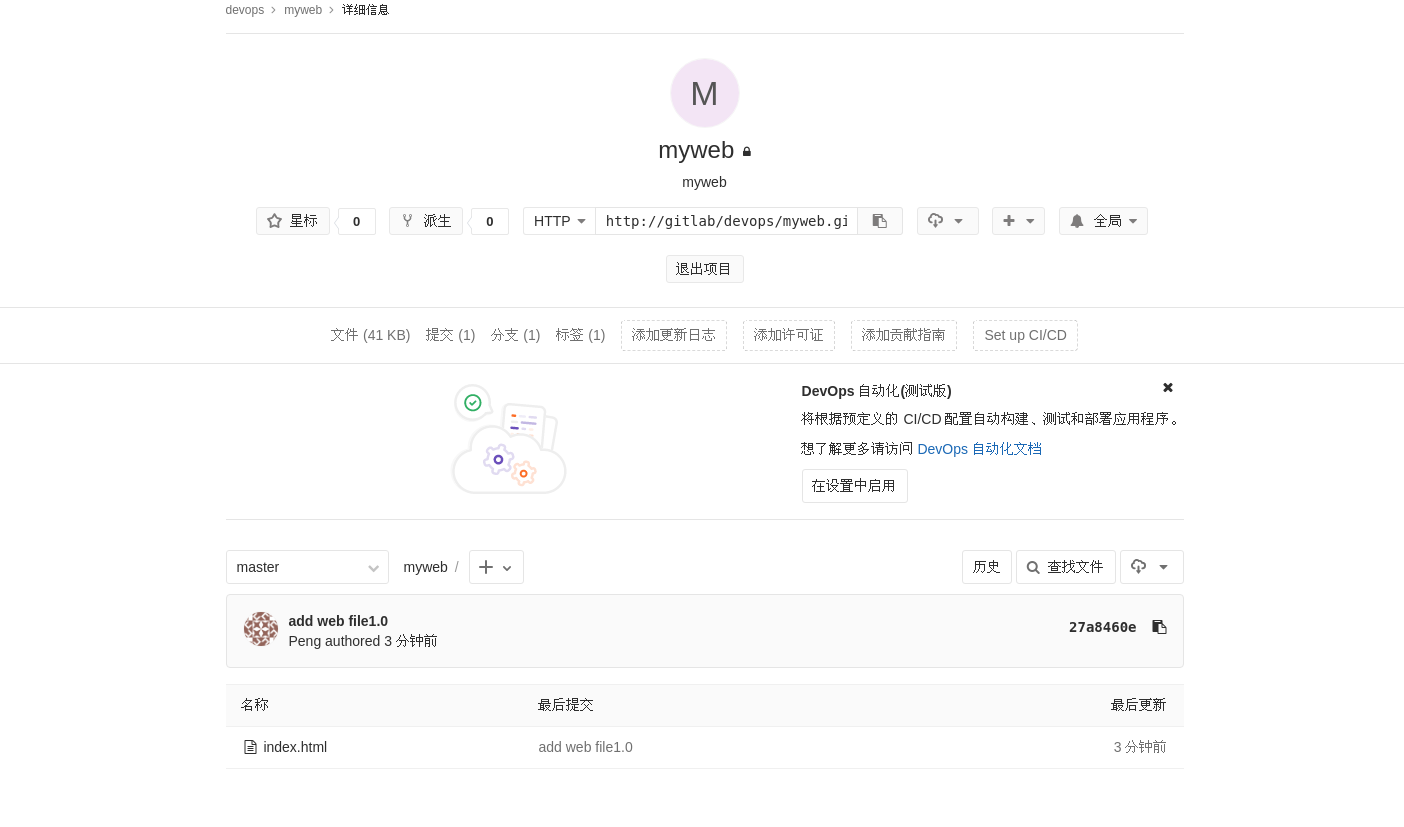
error: 不能重命名配置小节 'remote.origin' 到 'remote.old-origin'

[root@gcs-office myweb]# git remote add origin http://192.168.1.150/devops/myweb.git

[root@gcs-office myweb]# git push -u origin --all

[root@gcs-office myweb]# git push -u origin --tags

1. **Gitlab服务器上验证**



# 搭建jenkins服务

## 安装jenkins的依赖包java和jenkins

[root@jinkens ~]# yum -y install java-1.8.0-openjdk-devel

[root@jinkens ~]# yum -y install jenkins-2.187-1.1.noarch.rpm

## 启动服务，并查看

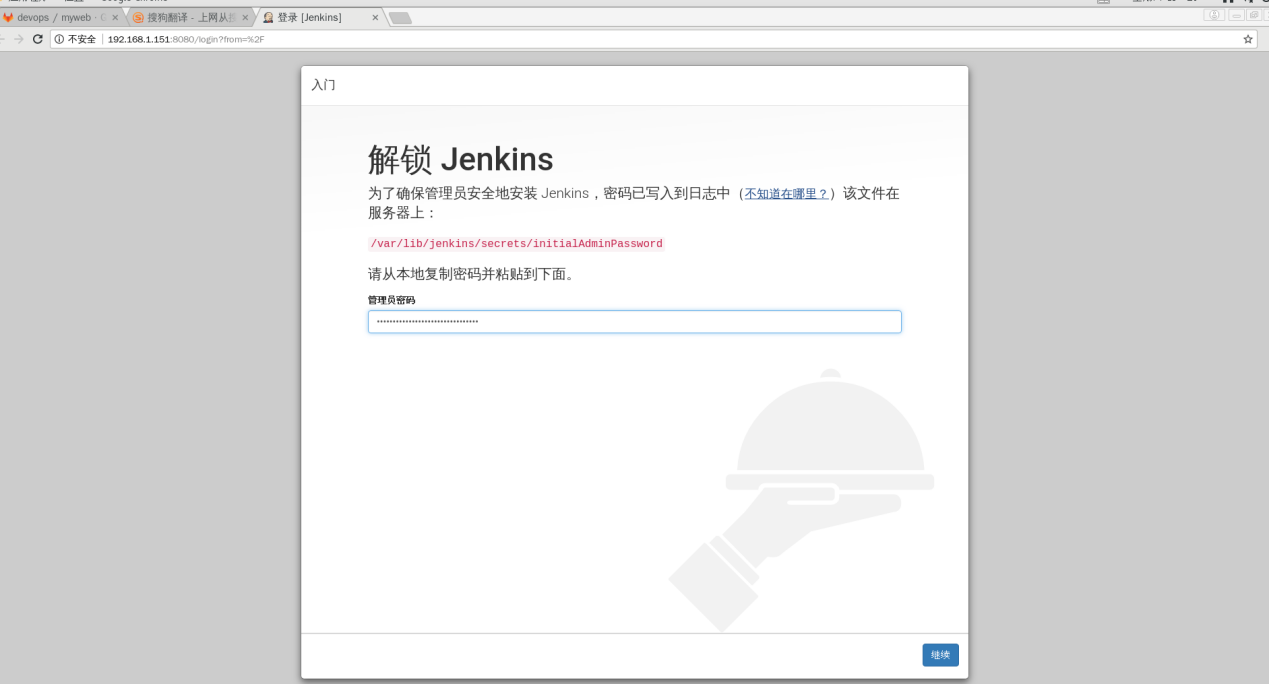
[root@jinkens ~]# systemctl start jenkins

[root@jinkens ~]# systemctl enable jenkins

[root@jinkens ~]# ss -nltup

## 初始化Jenkins

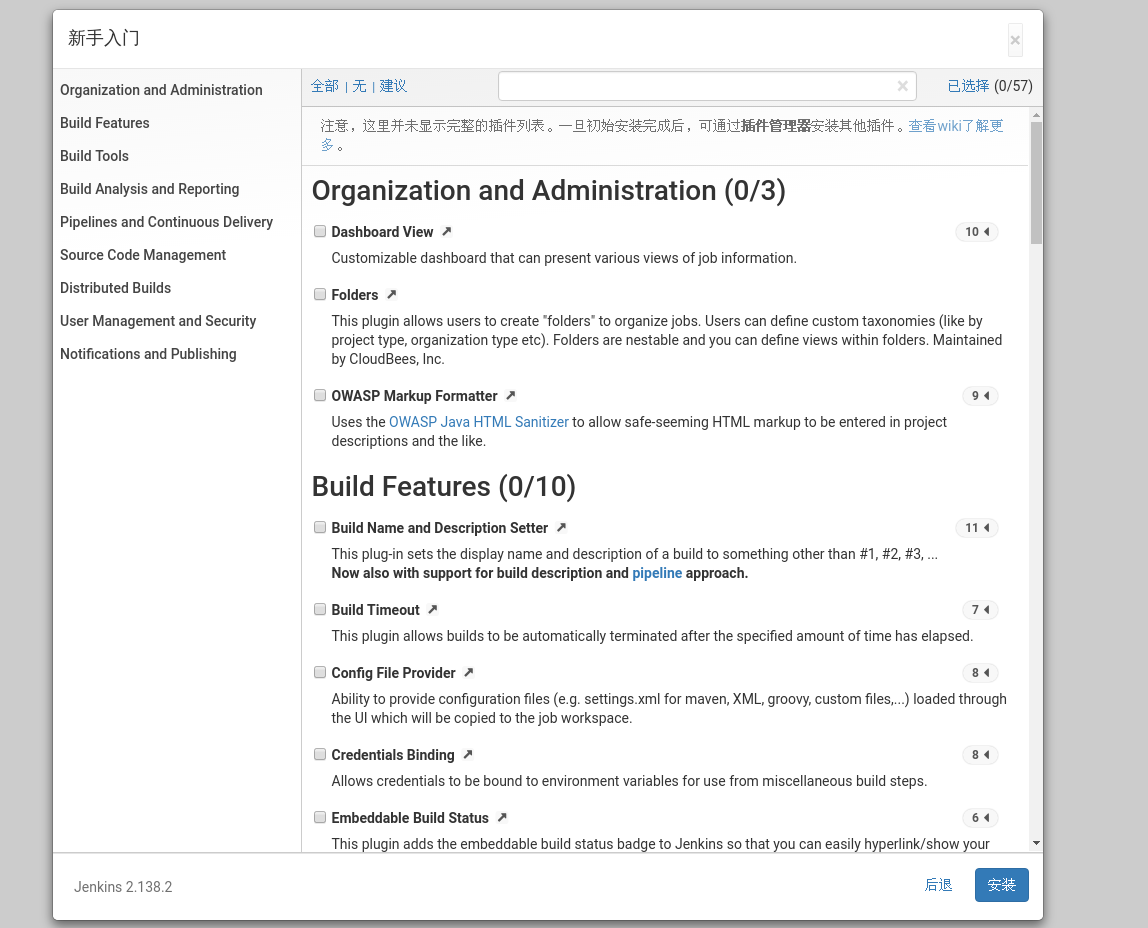
### 初始化



### 选择插件来安装



### 选择无来安装



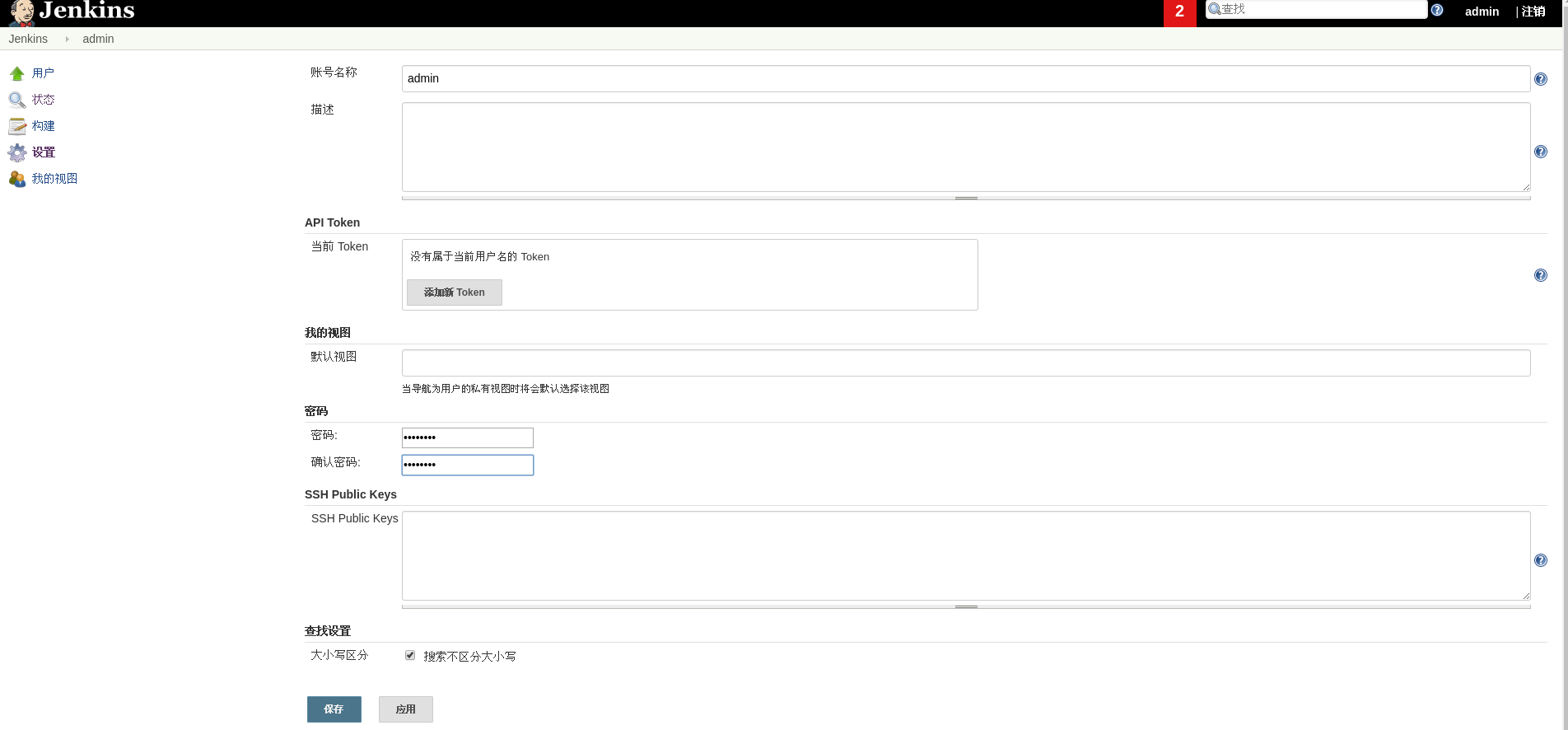
### 选择用admin用户



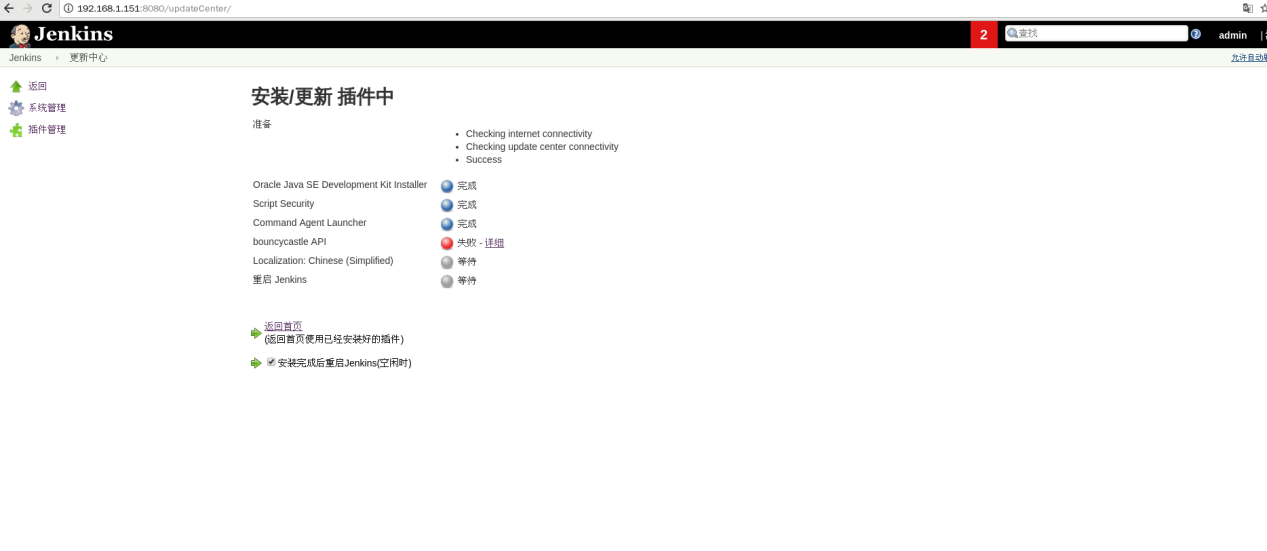
### 登陆并修改admin密码1234.com

http://mirror.xmission.com/jenkins/updates/update-center.json jenkins网站

https://mirrors.tuna.tsinghua.edu.cn/jenkins/updates/update-center.json 清华大学 jenkins站点



### 安装chinese中文与Git Parameter插件



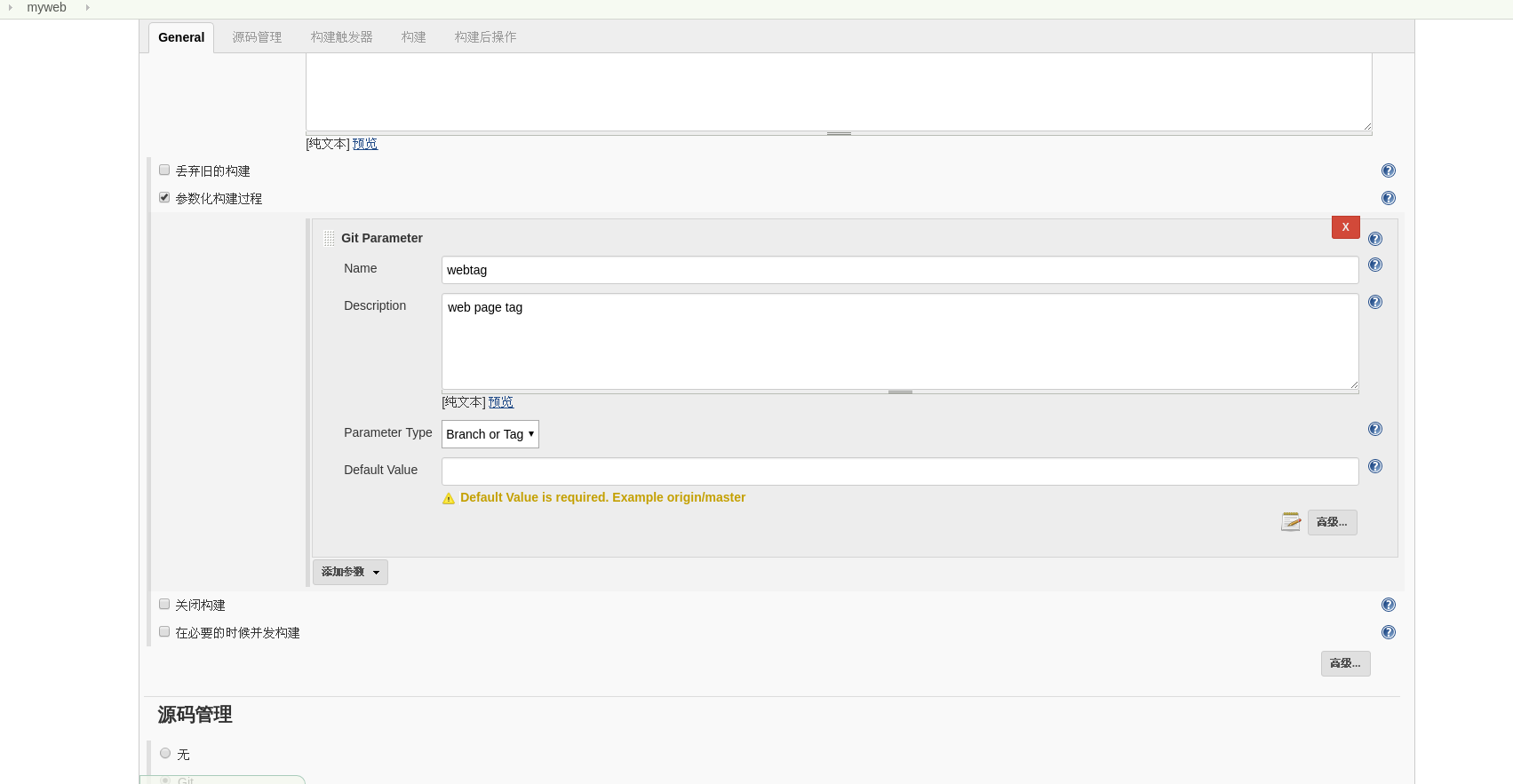


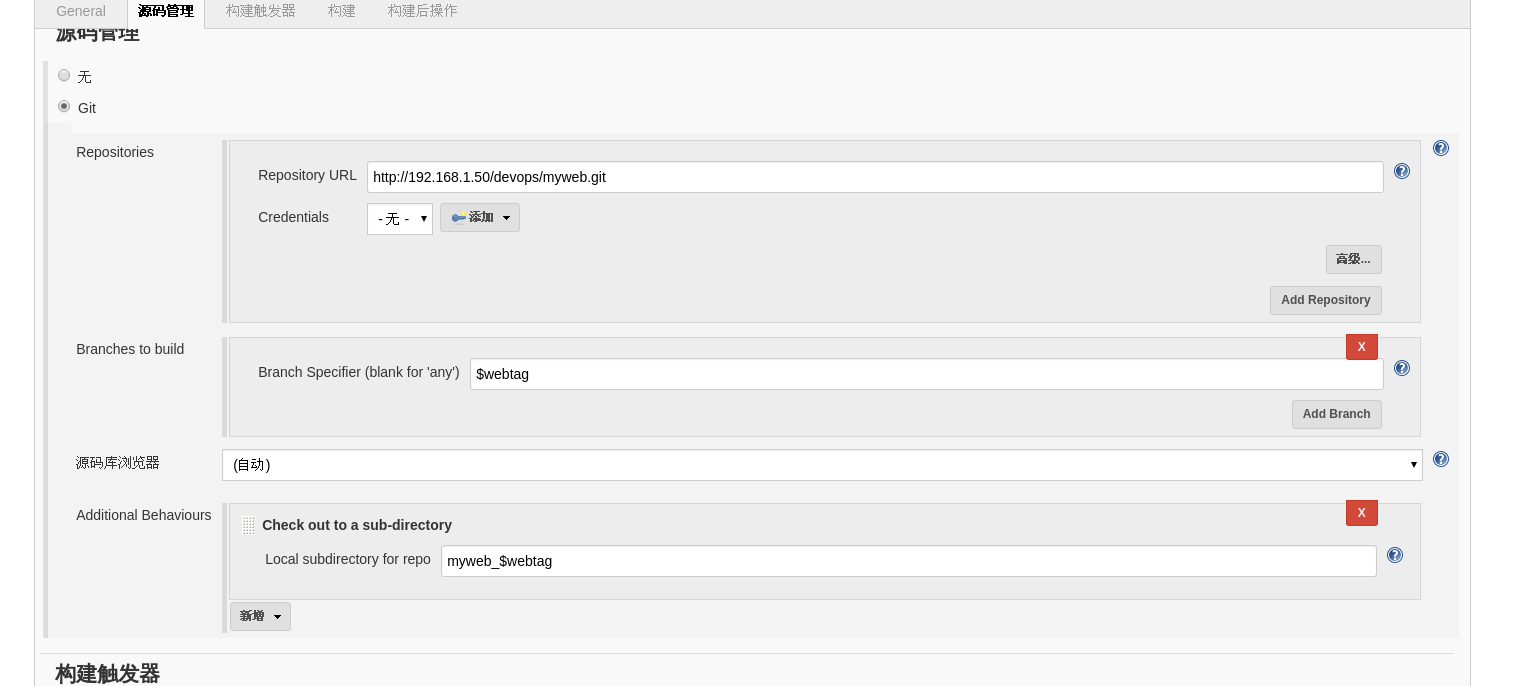
### 在jenkins创建一个项目

1.General-->参数化构建过程-->选择:git parameter-->(name:webtag)-->(description:web page tag)-->(parameter:branch or tag)

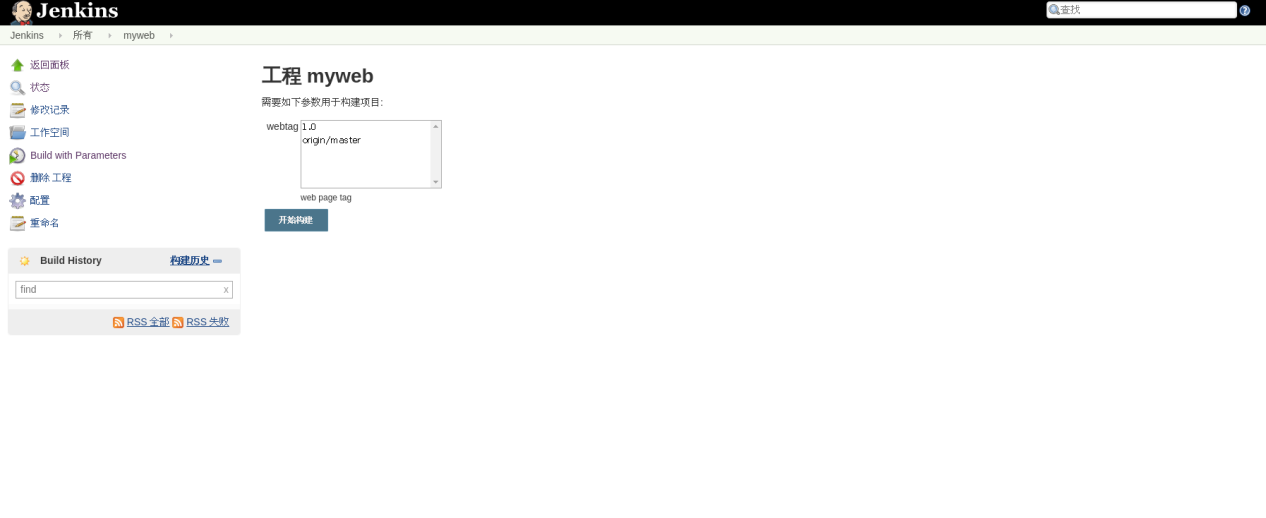
2.源码管理：git-->repositoy URL(http:git网站或者ssh)-->(Branch Specifier (blank for 'any':$webtag)

3.保存/应用





### 检查jenkins与gitlab服务器是否关联成功．





[root@jenkins myweb]# ls

myweb\_1.0 myweb\_2.0

[root@jenkins myweb]# pwd

/var/lib/jenkins/workspace/myweb

## 构建分发服务器管理

### Jenkins服务器上安装git和web网站

• 通过web服务为应用服务器提供应用程序

• 下载目录为/var/www/deploy/packages,并修改deploy目录的属组

[root@jenkins myweb]# yum -y install httpd git

[root@jenkins myweb]# mkdir -pv /var/www/html/deploy/packages

mkdir: 已创建目录 "/var/www/html/deploy"

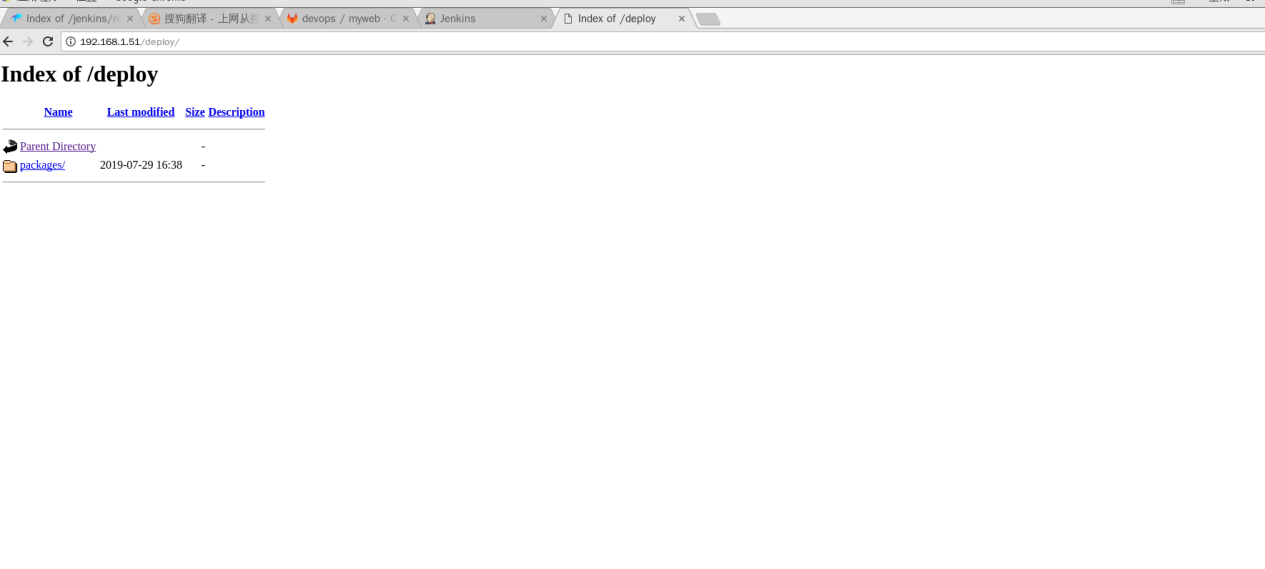
mkdir: 已创建目录 "/var/www/html/deploy/packages"

[root@jenkins myweb]# chown -R jenkins.jenkins /var/www/html/deploy/

[root@jenkins myweb]# ll /var/www/html/deploy/

总用量 0

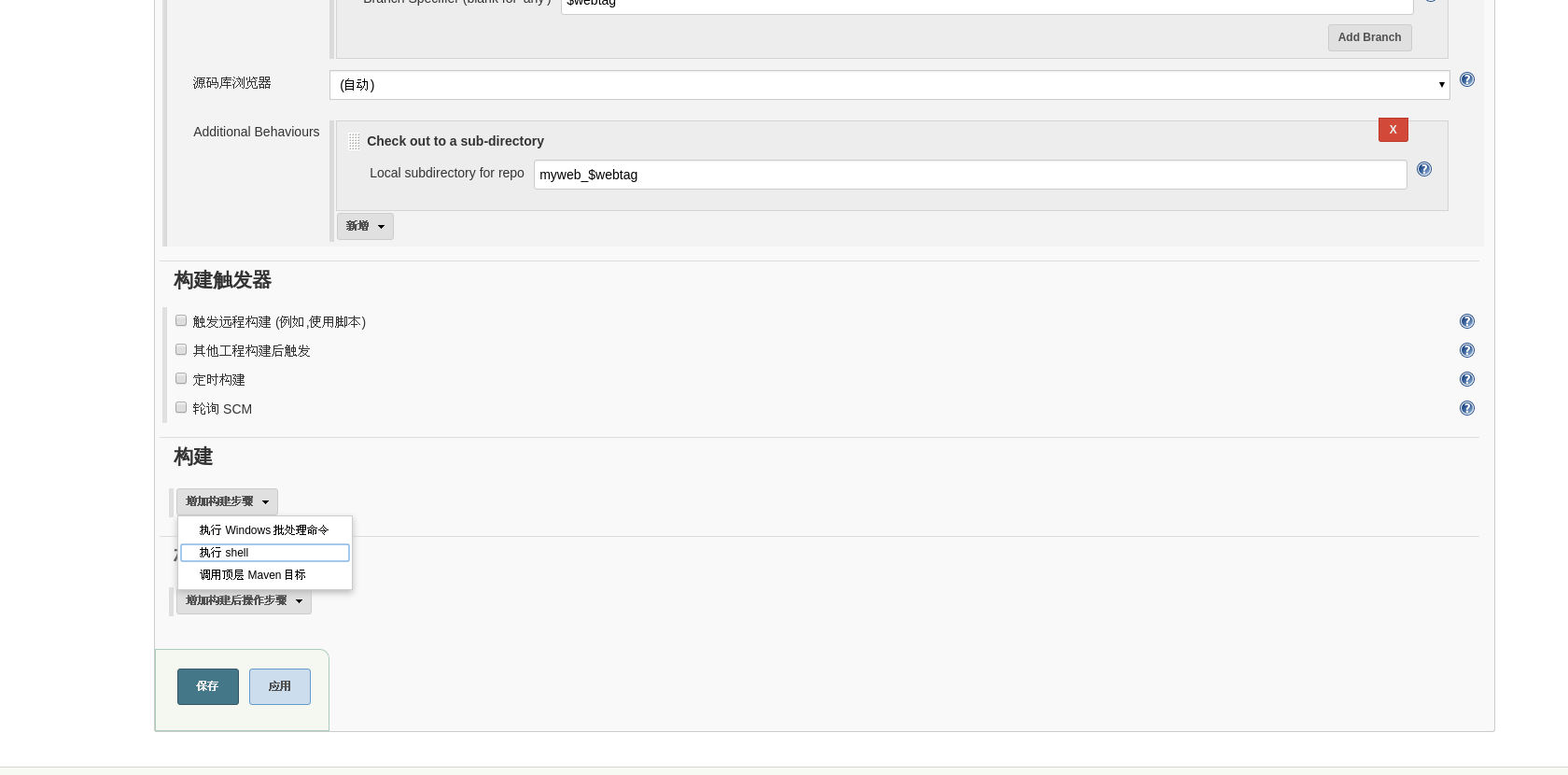
drwxr-xr-x 2 jenkins jenkins 6 7月 29 16:38 packages

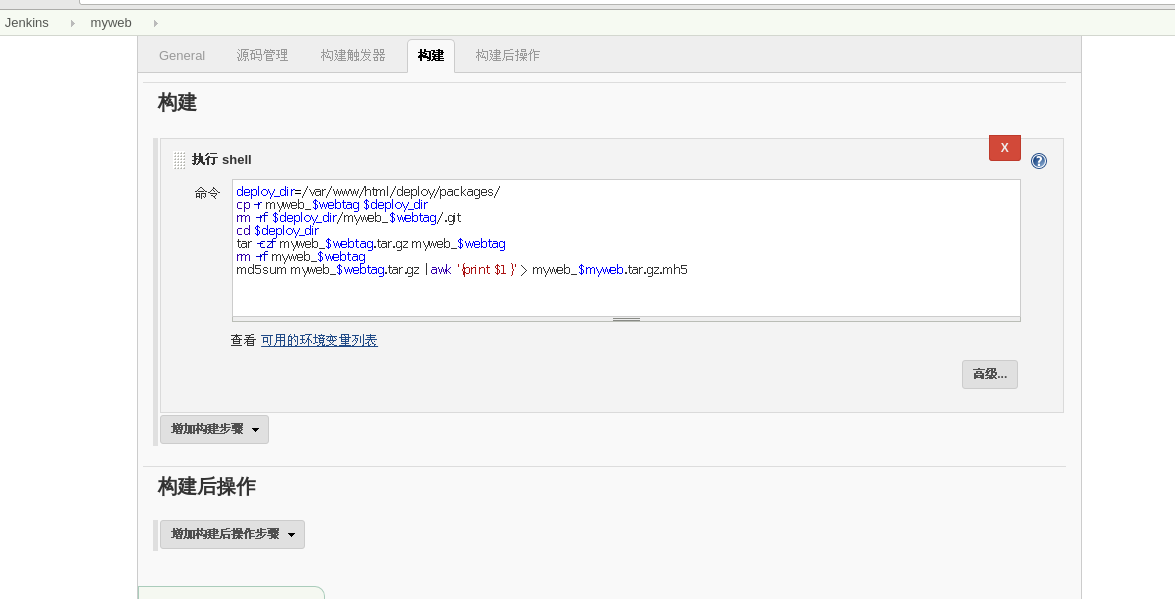


### 修改工程构建工程(续1)

• 为下载的应用打包,以及生成md5可以能过在工程中

增加构建步骤完成





deploy\_dir=/var/www/html/deploy/packages/

(定义一个是web网站的代码包的路径)

cp -r myweb\_$webtag $deploy\_dir

rm -rf $deploy\_dir/myweb\_$webtag/.git

cd $deploy\_dir

tar -czf myweb\_$webtag.tar.gz myweb\_$webtag

(将下载下来的文件打包)

rm -rf myweb\_$webtag

md5sum myweb\_$webtag.tar.gz | awk '{print $1}' > myweb\_$myweb.tar.gz.mh5

(将从gitlab服务器下载的包计算出md5值，并将其保存在一个文件中)

### 构建测试并查看jenkins服务上



[root@jenkins packages]# ls

myweb\_1.0 myweb\_2.0.tar.gz myweb\_.tar.gz.mh5

### 创建版本文件(版本文件的属组是jenkins)

• 创建两个版本文件

– live\_version:表示当前使用版本

– last\_version:表示上一个版本

• 应用服务器可以查看live\_version决定是不是要发布

新版本

• 如果新版本有问题,应用服务器可以根据

last\_version回滚到前一版本



[root@jenkins deploy]# touch livever

[root@jenkins deploy]# chown -R jenkins.jenkins livever

[root@jenkins deploy]# ll

总用量 0

-rw-r--r-- 1 jenkins jenkins 0 7月 29 17:14 lastver

-rw-r--r-- 1 jenkins jenkins 0 7月 30 10:37 livever

drwxr-xr-x 2 jenkins jenkins 58 7月 30 10:37 packages

### 服务器测试．

**环境准备：**

1. python的版本:python3.6
2. 在服务器创建网站一键部署目录’/var/www/deploy’使用root权限创建．
3. 创建网站代码版本文件’/var/www/deploy/livever’现在使用的版本
4. 使用root账号创建网站代码下载jenkins服务器上的目录’/var/www/download’
5. 使用root账号创建网站的首页网页文件’var/www/html/nsd1902’

### 网站一键部署代码编写．

**import** os  
**import** requests  
**import** wget  
**import** hashlib  
**import** tarfile  
  
**def** has\_new\_ver(ver\_url, ver\_fname):  
 **if not** os.path.isfile(ver\_fname):  
 **return True** r = requests.get(ver\_url)  
 **with** open(ver\_fname) **as** fobj:  
 local\_ver = fobj.read()  
  
 **if** r.text != local\_ver:  
 **return True  
  
 return False  
  
  
def** check\_app(app\_fname, app\_md5\_url):  
 m = hashlib.md5()  
 **with** open(app\_fname, **'rb'**) **as** fobj:  
 **while True**:  
 data = fobj.read(4096)  
 **if not** data:  
 **break** m.update(data)  
 result = m.hexdigest()  
  
 r = requests.get(app\_md5\_url)  
 **if** result == r.text.strip():  
 **return True  
  
 return False  
  
  
def** deploy(app\_fname, web\_root):  
 dest = **'/var/www/deploy'** tar = tarfile.open(app\_fname)  
 tar.extractall(path=dest)  
 tar.close()  
  
 app\_dir = app\_fname.split(**'/'**)[-1]  
 app\_dir = app\_dir.replace(**'.tar.gz'**, **''**)  
 app\_dir = os.path.join(dest, app\_dir)  
  
 **if** os.path.exists(web\_root):  
 os.remove(web\_root)  
  
 os.symlink(app\_dir, web\_root)  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 ver\_url = **'http://192.168.1.51/deploy/livever'** ver\_fname = **'/var/www/deploy/livever'  
 if not** has\_new\_ver(ver\_url, ver\_fname):  
 print(**'未发现新版本。'**)  
 exit(1)  
  
 r = requests.get(ver\_url)  
 version = r.text.strip()  
 app\_url = **'http://192.168.1.51/deploy/packages/myweb\_%s.tar.gz'** % version  
 download\_dir = **'/var/www/download'** wget.download(app\_url, download\_dir)  
  
 app\_fname = app\_url.split(**'/'**)[-1]  
 app\_fname = os.path.join(download\_dir, app\_fname)  
 app\_md5\_url = app\_url + **'.md5'  
 if not** check\_app(app\_fname, app\_md5\_url):  
 print(**'压缩包已损坏'**)  
 os.remove(app\_fname)  
 exit(2)  
  
 **with** open(ver\_fname, **'w'**) **as** fobj:  
 fobj.write(version + **'\n'**)  
  
 web\_root = **'/var/www/html/nsd1902'** deploy(app\_fname, web\_root)

**代码解析：**

1. 检查网站网页代码版本(现在使用的版本与jenkins服务器上最新的版本)．
2. 下载网站网页代码通过版本来控制(版本就是gitlab服务器中的tag便签)，为了下载方便所以所有代码都是通过\*.tar.gz包方式．下载下来的网站网页代码需要检查包是否完整，我们可以通过对比md5值，来确认下载的代码是否完整.
3. 最后一键部署，通过把下载完整的网站网页代码软链接到网站首页文件’/var/www/html/nsd1902’．

### 网站网页代码更新使用方案

1. 网站网页代码主程序员在通过git上传代码到gitlab服务器上．

[root@git-client myweb]# echo '<h3> hello world nsd1902 </h3>' >> index.html

[root@git-client myweb]# git add .

[root@git-client myweb]# git commit -m "add web page 3.0"

[master bb07ddb] add web page 3.0

1 file changed, 1 insertion(+)

[root@git-client myweb]# git tag 3.0

[root@git-client myweb]# git push

[root@git-client myweb]# git push origin --tags

1. 在jenkins服务器所在的网站myweb项目下查看网站网页代码主程序员更新的代码．



通过查看发现有新的代码发布．构建代码更新．

1. 手动执行代码．

(NSD1902) [root@room9pc01 day01]# python3 deploy.py

100% [..............................................................] 225 / 225((NSD1902) [root@room9pc01 day01]#

1. 访问网站查看一键部署结果．



### 网站网页代码应急方案

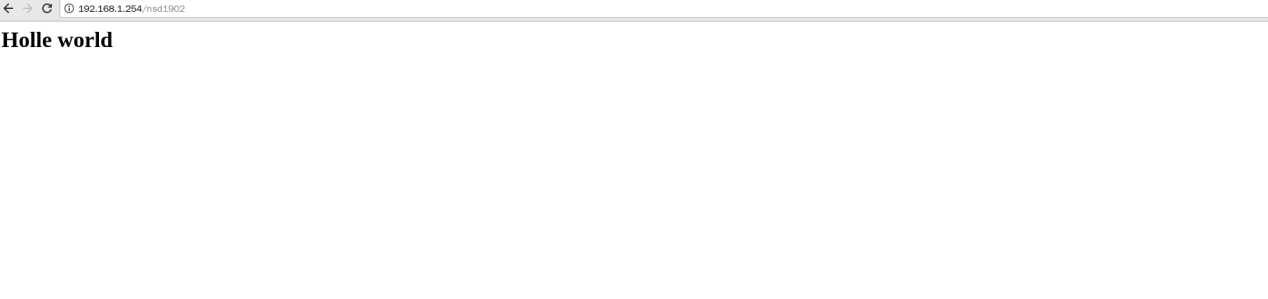
**按照不同网站的架构有不同的解决：**

1. 网站架构是单台web网站时．

方案是：删除网站网页的文件．将/var/www/deploy上次的网页代码软链接到要发布网站网页文件．

(NSD1902) [root@room9pc01 www]# rm -rf html/nsd1902

(NSD1902) [root@room9pc01 www]# ln -s /var/www/deploy/myweb\_1.0/index.html /var/www/html/nsd1902



1. 网站架构是web集群方式时，前置使用的是nginx反向代理时．

方案是：web前置服务器的nginx配置文件准备三份,里面的内容主要是使用upstream定义后端服务器集群分三份，location / {中通过proxy\_pass 将用户的请求转发给后端集群．